Introspection-An Optimal Technique for Dynamic Analysis of Object Oriented Programs

Mohit Kumar[1], Dr. Harsh Sadawarti[2], Dr. S.N. Panda[3]

A.P. CSE Dept RIMT-IET, Director RIMT-IET, Director RIMT-IMCT
MandiGobindgarh, India.

[1]Moiht.nabha@gmail.com , [2]harshsada@yahoo.com,[3] panda.india@gmail.com

Abstract

In last few years, Dynamic metrics was hot area of research in software engineering. A number of studies have explored how dynamic analysis can measure the quality of Object Oriented Software. Most of the studies utilized dynamic coupling, cohesion and complexity metrics to assess the quality of software. Unfortunately, the use of such metrics has not been practically validated. They were just implemented on small programs and their use for practical applications has so far not been thoroughly investigated. To clarify this problem, this paper presents a new approach for dynamic analysis of object oriented systems.

Keywords

Dynamic Metrics; Object Oriented Programming; Dynamic Analysis; Software Quality

Introduction

Towards moving 21st century, software hasbecome a driver for everything from elementary education to genetic engineering. Software developers analyze reliability, *-maintainability and complexity of software through software measurement. According to literature survey, high coupling decreases reliability of software and makes the software unreliable [Sarimbekov & Sewe (2010)]. System testing is concerned with testing an entire system based on its specifications, and involves several activities such as functional testing (testing from behavioral descriptions of the system) and performance testing (response time and resource utilization) [Briand & Labiche (2001)]. Coupling analysis is one of several techniques that model and measure the relationships among components in a software system, in which two components are coupled if any kind of connection or relationship exists between them. The coupling nature is often categorized into different levels or types. Coupling measurement has traditionally been performed using static code analysis, because most of the existing work was done on non-object oriented code and dynamic code analysis was more expensive and complex to perform. For modern software systems, however, this focus on static analysis can be problematic because although dynamic binding has existed before the advent of object orientation, its usage has increased significantly in the last decade [Tahir&Ahamd(2010)].

Dynamic coupling as the phenomenon in nature of objects binding together that create a state of balance is the fundamental law of attraction. There are different ways to define dynamic coupling, all of which can be justified, depending on the application context where such measures are to be used. Three decision criteria used to define and classify dynamic coupling measures are Entity of measurement, Granularity and Scope [Lavazza (2012)]. Dynamic analysis is the basic technique to collect data for the measurement of dynamic software metrics. Dynamic analysis of a software system involves the investigation of the properties of the software using information gathered from the run-time profiles or from dynamic models of the system. Typically, dynamic analysis of a program involves instrumentation of the program to record certain aspects of its dynamic state. Dynamic analysis is usually performed using program profiles and a program profile is used to count the number of times program entities take in a program execution [Gunnalan (2005)]. Measurement is recognized as a key element of any engineering process to assess the quality of an engineered product [Myao (2009)]. Measurement, which can be considered as mapping from the empirical world to the formal and relational world, is the number or symbol assigned to an entity in order to characterize an attribute [Gupta (2011)]. Coupling or dependency is the degree to which each program module relies on each one of the other modules [Mitchell (2003)]. Coupling measures capture

the degree of interaction and relationships among source code elements, such as classes, in software systems. Coupling is one of the fundamental properties of software with most influence on software maintenance and evolution [Prakashini (2010)]. In terms of internal validity, it is clear that coupling is only one of the factors affecting change proneness [Lavazza (2012)].

System classes that are strongly coupled are most likely to be affected by changes and bugs from other classes; and tend to have an increased architectural importance and thus they need to be identified. Coupling measures help in such endeavors, and most of them are based on some form of dependency analysis, or the available source code or design information [Parvathi(2012)]. A vast majority of coupling and cohesion metrics abound in the literature relies on structural information, which captures relations, such as method calls or attributes usages. These metrics have been proved useful in different tasks, such as, assessment of design quality, impact analysis, prediction of software quality, and faults, identification of design patterns etc [Yacoub(1999)]. Dynamic coupling measures have beenintroduced as the refinement to existing coupling measures due to gaps in addressing polymorphism, dynamic binding, and the presence of unused code by static structural coupling measures [Parvathi(2012)].

Even since the inception of object-oriented programming, class has been the primary unit of organization and reuse. However, over the years it has been acknowledged that a set of collaborating classes is a better unit of organization than a single class. Different names have been given to the set of classes such as class categories, clusters, subject areas, domains, subsystems and units [Mitchell (2005)]. Classes are the fundamental building blocks in Object Oriented software. A class defines new types and encapsulates state information in a collection of state variables, and has a set of behaviors implemented by methods in which those state variables are applied. Classes define types used to instantiate objects [Alexander (2004)] which are the run time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. Objected-oriented software products are different from earlier or conventional software products as they use procedures and subprograms [Gupta (2011)]. Object-oriented measurement has become an increasingly popular research area. This is substantiated by the fact that i) several different frameworks for coupling and cohesion and (ii) a large number of different measures for object-oriented attributes such as coupling, cohesion, and inheritance [Rabuet (2009)]. Despite the need for empirical research into large-scale Object Oriented systems, the majority of object-oriented metrics research has concentrated upon defining sets of structural metrics [Cartwright (2000)]. In order to understand Object Oriented software, software engineers need to create a well-connected representation of the classes that make up the system. Each class must be understood individually and, then, relationships among classes as well.

There is great interest in use of the object-oriented approach to software engineering. This is due to a variety of claims about how it may improve the development of software, including such factors as greater reusability and increased extensibility [Chidamber (1998)]. Object-oriented analysis applies object-modeling techniques to analyze the functional requirements for a system.

TABLE 1 STATIC VS DYNAMIC ANALYSIS OF SOFTWARE

S.No	Static Analysis of software	Dynamic Analysis of software
1	It is static in nature.	It is dynamic in nature.
2	It lacks in measuring object oriented features of Programs.	It effectively measures Object oriented features of Programs like inheritance, polymorphism, Dynamic memory allocation, coupling, cohesion and complexity.
3	It takes less time as compared to Dynamic Analysis of software.	It takes more time to perform dynamic analysis of a program
4	The results obtained are less accurate to predict the quality of software.	The results obtained are more accurate to predict the quality of software.
5	Tools are easily available to perform this analysis.	Not many tools are available to perform dynamic analysis.
6	It is performed on code.	It is performed while code is executed
7	Example: Consider class Cls1 calls eight methods (m1 to m8) of different classes once each. It measures the number of methods called. It is static measure.	Example: Class Cls1 calls a method m1 eight times form Class Cls2. It measure number of times method called. It can be measured at run time only. So it is Dynamic Analysis.

As the use of object-oriented design and programming is growing in industry, it has been observed that inheritance and polymorphism are used more frequently to improve internal reuse in a system and to facilitate maintenance [Lavazza (2012)]. One of the

main goals behind Object Oriented analysis and design is to implement a software system where classes have high cohesion and low coupling among them. These class properties facilitate comprehension activities, testing efforts, reuse, and maintenance tasks [Yacoub (1999)]. Object-oriented modeling makes it possible to create physically relevant and easy-to-use model components, which are employed to support hierarchical structuring, reuse, and evolution of large and complex models covering multiple technology domains [Greenwood (2007)]. Many coupling measures have been proposed in the context of object-oriented systems. In addition, several studies have highlighted the complexity of using dependency analysis in object-oriented software to perform impact analysis [Skruch (2011)].

Motivations of the Research

For a long time, though dynamic analysis of programs has been a subject of study in the context of compiler optimization, it is the basic technique tocollect data for the measurement of dynamic software metrics. The results obtained in measuring dynamic metrics using dynamic analysis have been quite encouraging. There are several methods available for the coupling measurement in Object Oriented System; they are 1) Dynamic Coupling 2) Static Coupling 3) Package Coupling 4) Conceptual Coupling and so on. In this, dynamic and static analyses are complementary techniques in a number of dimensions. Dynamic analysis may be termed as "input-centric" analysis, whereas static analysis may be better termed as "program-centric" analysis. Since dynamic analysis is always dependent on inputs supplied to the program under consideration. The very thing that makes dynamic analysis incomplete also provides a powerful mechanism to relateprogram inputs and outputs to program behaviour [Gunnalan (2005)].

Conceptual coupling based on the semantic information shared between elements of the source code can be classified as measuring the strength of conceptual similarities among methods of different classes. The measures are based on using information retrieval (IR) techniques to model and analyze the semantic information embedded in software (i.e., through comments and identifiers). The conceptual coupling can be used to augment existing measures, especially in tasks such as impact analysis and change propagation, as existing models do not capture all the ripple effects of changes in existing software. They also have direct application in reverse engineering tasks like re-modularization [Mitchell (2005)]. These metrics have been proved useful in different tasks, such as, assessment of design quality, impact analysis, prediction of software quality, and faults, identification of design patterns etc [Yacoub (1999)].

Coupling between packages is the manner and degree of interdependence between them. Packages may consist of classes and sub-packages as their elements. Further, these sub-packages may contain classes and sub-packages as their elements. This leads to a hierarchical structure of packages in a software system. Theoretically, every package is a stand-alone unit, but in reality packages depend on many other packages as either they require services from other packages or they provide services to other packages. Thus, coupling between packages cannot be completely avoided but it can only be controlled. The coupling between packages is an important factor that affects the quality or other external attributes of software, e.g., reliability, maintainability, reusability, fault-proneness etc [Mitchell (2005)].

Coupling measurement has traditionally been performed using static code analysis, because most of the existing work was done on non-object oriented code and dynamic code analysis is more expensive and complex to perform. For modern software systems, however, this focus on static analysis can be problematic because although dynamic binding has existed before the advent of object orientation, its usage has increased significantly in the last decade. Statically analyzing code helps in monitoring whether it meets uniform expectations onsecurity, reliability, performance, and maintainability [Tahir (2010)]. Static analysis is restricted in analyzing a program effectively and efficiently, and may have trouble in discovering all dependencies present in the program [Gunnalan (2005)].

Dynamic coupling measures are collected through dynamic analyses which show that these measures are better predictors of quality attributes and better indices of complexity than traditional coupling measures [Tahir (2010)]. Dynamic analysis is usually performed using program profiles and a program profile is used to count the number of times program entities takein a program execution. Profiles can be recorded at many different levels, from that of objects, methods and procedures, down to paths, branches and even individual program instructions. Dynamic analysis withthe benefit of examining the concrete domain of program execution examines actual program executions, and is free from the problem of

examination of infeasible paths that can badly affect static analyses [Gunnalan (2005)].

Systematic Reviews

This section describes the objectives and questions as well as the strategically steps carried out in the systematic review.

Objectives and Questions

The aim of our systematic review is to analyse the effectiveness of dynamic metrics to access the quality of object oriented software. And particular focus is on the following four research questions:

a) Which external attributes are most frequently used to indicate quality of software?

b) Why dynamic analysis techniques are available and what are their advantage and disadvantages?

c) Does dynamic analysis effectively measure these attributes?

Review Strategy

Searches for papers took place in 20 renowned online journal banks or were those published in recognized conference papers. Conferences with an acceptance rate below 25% given priorities. Relevant papers were found from ACM, SpringerLink, IEEE, Google Scholar, and collected from other sources.

Sampling Criteria: From this base, we sampled papers that met the following criteria. Each selected paper has to:

- Use an empirical study of object oriented systems.
- And use dynamic metrics within the study.

Due to low retrieval rate from journal banks, alternative approaches were also used including both consulting references on already-found papers and searching specifically for papers we knew met our criteria (from previous knowledge). The distribution of collected research is recorded in the review results.

Results

TABLE 2 DISTRIBUTION OF STUDIES

Electronic Journal	Retrieved	Rejected	Used
ACM	8	3	5
IEEE	7	3	4
SpringerLink	6	1	5
Others	15	5	10
Total	**36**	**12**	**24**

A final set of 24 papers was finally obtained (Table 2), which is a typical sample sizefor systematic reviews in software engineering [Greenwood (2007)].

Attributes of software Quality: IEEE ISO 9126 provides a listing of Software Quality Characteristics and Attributes. The eight characteristic groups proposed in ISO 9126 are:

Functionality. Characteristics related to achievement of the basic purpose for which the software is engineered.

Reliability. Characteristics related to the capability of software to maintain its level of performance under stated conditions for a stated period of time.

Understandability. Characteristics related to the effort needed for use and on the individual assessment of such use, by a stated or implied set of users.

Efficiency. Characteristics related to the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

Maintainability. Characteristics related to the effort needed to make modifications, including corrections, improvements or adaptation of software to changes in the environment, requirements and functional specifications.

Portability. Characteristics related to the ability to transfer the software from one organisation or hardware or software environment to another.

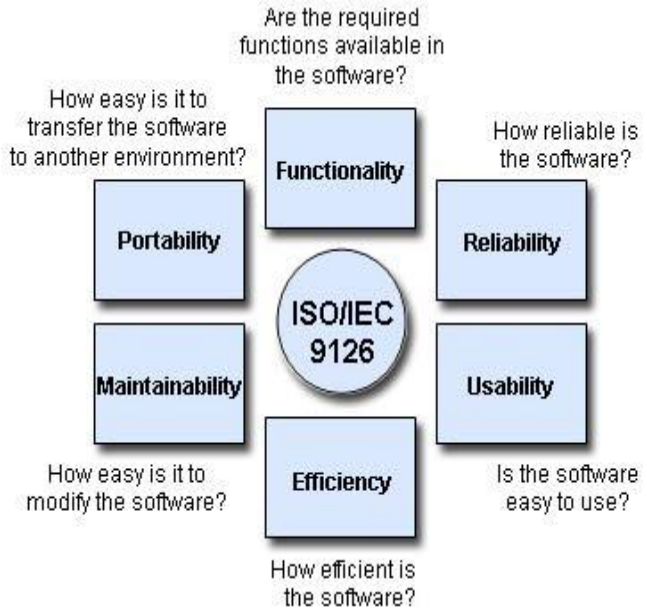

FIG. 1 SOURCE: SOFTWARE ENGINEERING BODY OF KNOWLEDGE, P11-4.

http://www.fe.up.pt/~jpf/teach/ES/Guide_to_the_SWEBOK-Stone_Man_Version_100.pdf

Available Dynamic Analysis Techniques, Their Advantage and Disadvantages

TABLE 3 VARIOUS DYNAMIC ANALYSIS TECHNIQUES AVAILABLE:-

S. No.	Dynamic Analysis Technique	Advantages	Disadvantages
1	Dynamic Modeling Diagrams	Low cost on program counters. Valid for all programming environments. Provides all runtime information	Excessively cautions. Large amount of data is generated Unused code is considered
2	Dynamic Analysis Based On Profilers	Beneficial in assessing critical system Most Efficient for small Applications	Complex Generates Large data Takes more time Results depends upon input strategy Not suitable for Large applications
3	Dynamic Analysis Using Aspect-Oriented Approach	Simple Less data is generated Inexpensive Results valid for all execution environments	Programming skills are required. Manual Join points are created
4	Preprocessor approach for dynamic analysis	Preprocessor is used Easy to use	Costly Conflict with other preprocessors Maintainability and reliability issues
5	AST rewriting based analysis	Based on Abstract Syntax tree	Not commonly used.
6	Ad-hoc approaches for dynamic analysis	Manual scripting is done Efficient for small application	Not a better choice for large applications
7	Hybrid Approach	Combination of static and dynamic approaches	Verification is required

Efficiency of Dynamic Analysis Techniques

From the above section, it is clear that there are not enough dynamic analysis techniques available to measure quality attributes of software. Meanwhile, there is a large scope of dynamic analysis of software.

Literature Survey

The majority of the measures for internal software attributes (e.g., structural size, structural complexity, cohesion, and coupling) have been defined in a static way. But statically examining the potential interactions between a software module and the rest of software system may not be sufficient, for example in assessing the coupling of the module. Suppose that module A has fewer static connections with the rest of the system than module B does, but that module A at run-time interacts with the rest of the system much more frequently than B does. So, B may be considered to have a higher coupling than A from a static perspective, while the converse may be true from a dynamic point of view. As another example, static size measurement does not capture the difference between heavily used and lightly used software code and may even take into account dead code. Again, when estimating the number of faults found in a routine, dynamic size may be more relevant than static size. Thus, it should be investigated whether dynamic measures may be used alongside static measures in the assessment of some software quality of interest or even if dynamic measures may be more useful than static ones. The distinction between static and dynamic internal software measures parallels, from a practical point of view, the distinction between static and dynamic analysis in software verification and validation. Both kinds of analysis may be useful when looking for software faults or when assessing some non functional properties, like maintainability, performance, or usability. To measure dynamic properties of asoftware Luigi Lavazza, SandroMorasca, DavideTaibi and DavideTosi have proposed dynamic measure for size and coupling. Under size measures, they proposed five metrics as instructions-the number of byte code instructions has been exercised, SLOC - numberof Java source lines of code exercised, Methods-the number of distinct methods exercised and type-the number of Java type exercised. And for coupling they proposed three metrics as distinct method invocations, Dynamic messages and distinct classes. [Lavazza (2012)]. Claudia Raibulet, Laura Masciadri proposed metrics for the evaluation of runtime adaptively. The advantage of such metrics is the specification of a common vocabulary for different design, implementation, and performance adaptively, and provided a common means for the evaluation of adaptive systems by considering both the quality of their design (through the architectural and structural metrics) and of their performances (through the interaction and performance metrics). From the software engineering point of view, the metrics belonging to the architectural category may be exploited during the analysis and design phases. The structural metrics represent useful hints for the

implementation phase. The interaction and performance metrics are particularly meaningful in the testing phase. Furthermore, the architectural and structural metrics may be exploited during the maintainability decisions because they provide information about the quality of the design of a system [Rabuet (2009)]. PawelSkruch from AGH University of science and technology Poland presents a test of quality measure that allows for quantifying the completeness of black-box tests for continuous-time dynamic systems. The measure is based on a state space model of the system under test. The metric has been called the state space coverage. The classical coverage metrics, such as statement, branch, and path coverage, are not appropriate for dynamic systems because such systems are defined by differential equations and usually have an infinite number of states. An application example has been presented to illustrate theoretical analysis and mathematical formulation [Skruch (2011)]. Prakashini presented obtainable and new software metrics useful in the different phase of the Object-Oriented Software Development Life Cycle. These metrics are used by the software industry to itemize the development, operation and maintenance of software. The practice of applying software metrics to a software process and to a software product is a complex task that requires study and restraint, which brings knowledge of the status of the process and or product of software in regards to the goals to be, achieved [Prakashini (2010)]. Sherifyaccoub& Tom Robinson addressed the problem of measuring the quality of object-oriented designs using dynamic metrics and presented a metrics suite to measure the quality of designs at an early development phase. This suite consists of metrics for dynamic complexity and object coupling based on execution scenarios. The proposed measures are obtained from executable design models. They apply the dynamic metrics to assess the quality of a "Pacemaker" application. And results from the case study are used to compare static metrics to the proposed dynamic metrics and hence identifying the need for empirical studies to explore the dependency of design quality on each [Yacoub (1999)]. Aine Mitchell submitted a thesis for the degree of Doctor of Philosophy in which the extent of coupling and cohesion in an object-oriented system has implicated for its external quality, mentioned that various static coupling and cohesion metrics have been proposed and used in past empirical investigations, but none of these have taken the run-time properties of a program into account, and believed that any measurement of these attributes should include changes that take place at run-time. For this reason, he addressed the utility of run-time coupling and cohesion complexity through the empirical evaluation of a selection of run-time measures for these properties. This study was carried out using a comprehensive selection of Java benchmark and real world programs. His first case study investigated the influence of instruction coverage on the relationship between static and run-time coupling metrics and second case study determined a new run-time coupling metric that can be used to study object behavior and investigated the ability of measures of run-time cohesion to predict such behavior [Mitchell (2005)]. Dr. R.M.S. have proposed a new approach to the computation of dynamic coupling measures in DOO systems by introspection and adding trace events into methods, and proposed dynamic coupling measures for distributed object-oriented systems i.e., coupling measurement on both clients and server dynamically, as well described the classification of dynamic coupling measures. The motivation for those measures is to complement existing measures that are based on static analysis by actually measuring coupling at runtime in the hope of obtaining better decision and prediction models for the sake of precise inheritance, polymorphism and dynamic binding [Parvathi(2012)]. Hany H. Ammar provided the concept of Pseudo Dynamic Metricsthat are static metrics adjusted to reflect the operational profile of the expected usage. Pseudo dynamic metrics can be estimated earlier than dynamic metrics as they depend only on the static metrics and operational profiles, which can be obtained from UML specifications [Gunnalan (2005)].

Although lots of dynamic metrics have been proposed in the literature to measure coupling, cohesion and complexity, but very few metrics are designed to measure other object oriented features like polymorphism, dynamic binding, dynamic memory allocation etc. [Lavazza (2012)].And impact of these metric on external software attributes like understandability, maintainability, reusability and reliability has yet been defined.

In this study, it was also found that dynamic metric are measured in last stage of software life cycle model where impact of metric measured in early stages of software life cycle is significant because the changes required can be implemented more efficiently at this time of development. This is possible with the concept of pseudo dynamic metrics by which, the dynamic behavior of the software system can be captured earlier in the development cycle than, that using dynamic metrics. It was considered that it is possible to

establish statistical and even functional relationships between some static software metrics and their dynamic counterparts, after the static metrics are adjusted to reflect the operational profile of the expected usage. Our survey has further pointed out that little work has been done to use the dynamic metrics in real world applications.

Problem Definition

The static metrics are able to measure various aspects of software system, like coupling cohesion and complexity at design or source code level. But their ability to accurately predict the dynamic behavior of an application is not proven. Traditional static metrics alone may be insufficient to evaluate the dynamic behavior of an application at runtime, as its behavior will be influenced by the execution environment as well as the complexity of the source code. Object-oriented features such as polymorphism, dynamic binding, inheritance and common presence of unused code in software, render the imprecise static metrics, as they do not precisely reflect the runtime situation of the software. For example, polymorphic call depends on the run time type of the object receiving the call. In programs that employ inheritance, this target may change. Moreover, the complex dynamic behavior of many real-time applications motivates us to focus on dynamic metrics in place of traditional static metrics. Due to estimation on software quality attributes based on dynamic analysis, the software system will be more accurate and realistic.

In dynamic analysis, the dynamic behavior of the software system is usually obtained from the execution traces of the code or from the executable models. Major dynamic analysis metrics proposed have been for the measurement of coupling, cohesion, and complexity from code. The importance of measuring dynamic behavior of software to establish a relationship between static and dynamic metrics of software is need of time. To maximize the impact, dynamic metric, should be measured in early stages of software development process. This is possible with the help the concept of pseudo dynamic metrics, by which estimate of the dynamic behavior of software can be assessed in early stages of the software development lifecycle. In this study, the various dynamic metrics available in literature have been observed and they are relatedto static metrics so that a direct relationship between static and dynamic metrics can be established to find the software quality attribute in early stage of software development life cycle.

This research carried out attempts to narrow the gap between static metrics and dynamic analysis, and lay the foundation for a more systematic approach to estimate the dynamic behavior of a software system.

Hypothesis:

In software development projects and maintenance, the software metrics for testability and quality plays an important role. The attributes such as fault proneness, changeability, reliability and maintainability are used to predict the software quality in terms of software structural complexity. Coupling and cohesion arethe major research areas in software maintenance. In order to overcome the issues of existing software testability and quality prediction system, an efficient method for dynamic analysis and quality prediction would be developed based on soft computing techniques.

In this proposed methodology, the dynamic analysis will be measured based on the optimal introspection procedure which will identify dynamic functions and methods in the software. This will be done by Genetic Algorithm (GA). After identifying the coupling dynamic methods, the reliability and maintainability can also be calculated based on ISO 9126 Fig 1. Based on these three metrics, the quality of the software will be predicted. The proposed method will be implemented in JAVA in Object oriented systems.

Conclusion

It can be concluded that dynamic analysis hasedge over static analysis. This survey confirms that existing dynamic analysis techniques are not enough and there is ambiguity in finding suitable metrics to measure quality attribute.

Additionally, the question of validity of results of these techniques also arises. Dynamic analysis can be performed on the bases of optimal introspection technique.

A methodology can be developed to implement this technique in real world object oriented software applications, as well a relationship can be established between dynamic metrics and external quality attributes.

REFERENCES

Al Dallal, Jehad “Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics” IEEE Transactions on Software Engineering 2010 DOI 10.1109/TSE.2010.97.

Alexander, Roger T. and Offutt, Jeff, "Coupling-based Testing of O-O Programs," Journal of Universal Computer Science, Vol. 10, No. 4, pp. 391 - 427, 2004.

Antonellis, P., Antoniou, D., Kanellopoulos, Y., Makris, C., Theodoridis, E., Tjortjis, C. and Tsirakis, N.," Clustering for Monitoring Software Systems Maintainability Evolution", Journal of Electronic Notes in Theoretical Computer Science (ENTCS), Vol. 233, Mar 2009.

Briand, Lionel and Labiche, Yvan, "A UML-Based Approach to System Testing," In Proc. of the 4th International Conference on The Unified Modelling Language, Modeling Languages, Concepts, and Tools, London, 2001.

Cartwright, Michelle and Shepperd, Martin, "An Empirical Investigation of an Object-Oriented Software System," IEEE Transactions on Software Engineering, Vol. 26, No. 8, pp. 786 - 796, Aug 2000.

Chidamber, Shyam R., Darcy, David P. and Kemerer, Chris F., "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis," IEEE Transactions on Software Engineering, Vol. 24, No. 8, pp. 629 - 639, Aug 1998.

Emanuel, AndiWahjuRahardjo, Wardoyo, Retantyo and Istiyanto, JaziEko,"Modularity Index Metrics for Java-Based Open Source Software Projects",International Journal of Advanced Computer Science and Applications, Vol. 2, No. 11, 2011.

Greenwood, P. et al.: On the Impact of Aspectual Decompositions on Design Stability: An

Gunnalan, R., Hany, M.S, Ammar, H. "Pseudo Dynamic Metrics" 0-7803-8735-X/05/©2005 IEEE.

Gupta, V. "Validation of Dynamic Coupling Metric for Object Oriented Software" ACM SIGSOFT Software Engineering Notes September 2011Volume 36 Number 5.

Lavazza, Luigi, Morasca, Sandro "On the definition of dynamic software measures" ESEM-12 Lund Sweden 2012 ACM copyright. Empirical Study. ECOOP (2007) 176-200

Mitchell, Aine & James, F. "Toward a definition of run-time object-oriented metrics" Power 7th ECOPP workshop on Quantitative approaches in object-oriented software Engineering 2003.

Mitchell, Aine "An empirical study of run-time coupling and cohesion software metrics" Thesis for the degree of Doctor of Philosophy Oct 2005.

Myao, Kevin A., Wake, Steven A. and Henry, Sallie M. "Static and Dynamic Quality Metric Tools".

Parvathi, M.S., "Development of Dynamic Coupling Measurement of Distributed Object Oriented Software Based on Trace Events" International Journal of Software Engineering & Applications (IJSEA) Vol.3, No.1 January 2012.

Prakashini, C., "Metrics for Object Oriented Design Software Systems: A Survey" Journal of Emerging Trends in Engineering and Applied Sciences (JETEAS) 1 (2): 190-198. (c) Scholarlink Research Institue Journals, 2010 (ISSN:214-7016)

Rabuet, Claudia & Masciadri, Laura "Evaluation of Dynamic Adaptability through Metrics: an Achievable Target" 978-1-4244-4985-9, IEEE 2009.

Rizvi, S. W. A. and Khan, R. A., "Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD)",Journal Of Computing, Vol. 2,No .4,pp.26-32, Apr 2010.

Sarimbekov, Aibek&Sewe, Andreas, "JP-2 Collecting Dynamic Byte code Metrics in JVMs SPLASH'11 Oct 2011 Oregon, USA ACM 978-1-4503-0940-0/11/10.

Shatnawi, Raed and Li, Wei,"An Empirical Assessment of Refactoring Impact on Software Quality Using a Hierarchical Quality Model", International Journal of Software Engineering and Its Applications, Vol. 5, No. 4, Oct 2011.

Skruch, Pawel "A coverage metric to evaluate tests for continuous-time dynamic systems" Research artcle from AGH University of Science and Technology Poland, Printed in Central European Journal of Engineering January 2011 Cent. Eur.J. Eng.1(2).2011.174-180.

Tahir, Amjed&Ahamd, Rodina "An AOP-Based Approach for collection Software maintainability Dynamic Metric" 978-0-7695-4043-6/10, IEEE 2010. www.sdjournal.org .

Yacoub, Sherif, Robinson, Tom, and Ammar, Hany H. "Dynamic Metrics for Object Oriented Designs" Published in: Proceeding METRICS '99 Proceedings of the 6th International Symposium on Software Metrics Page 50 IEEE Computer Society Washington, DC, USA ISBN: 0-7695-0403-5